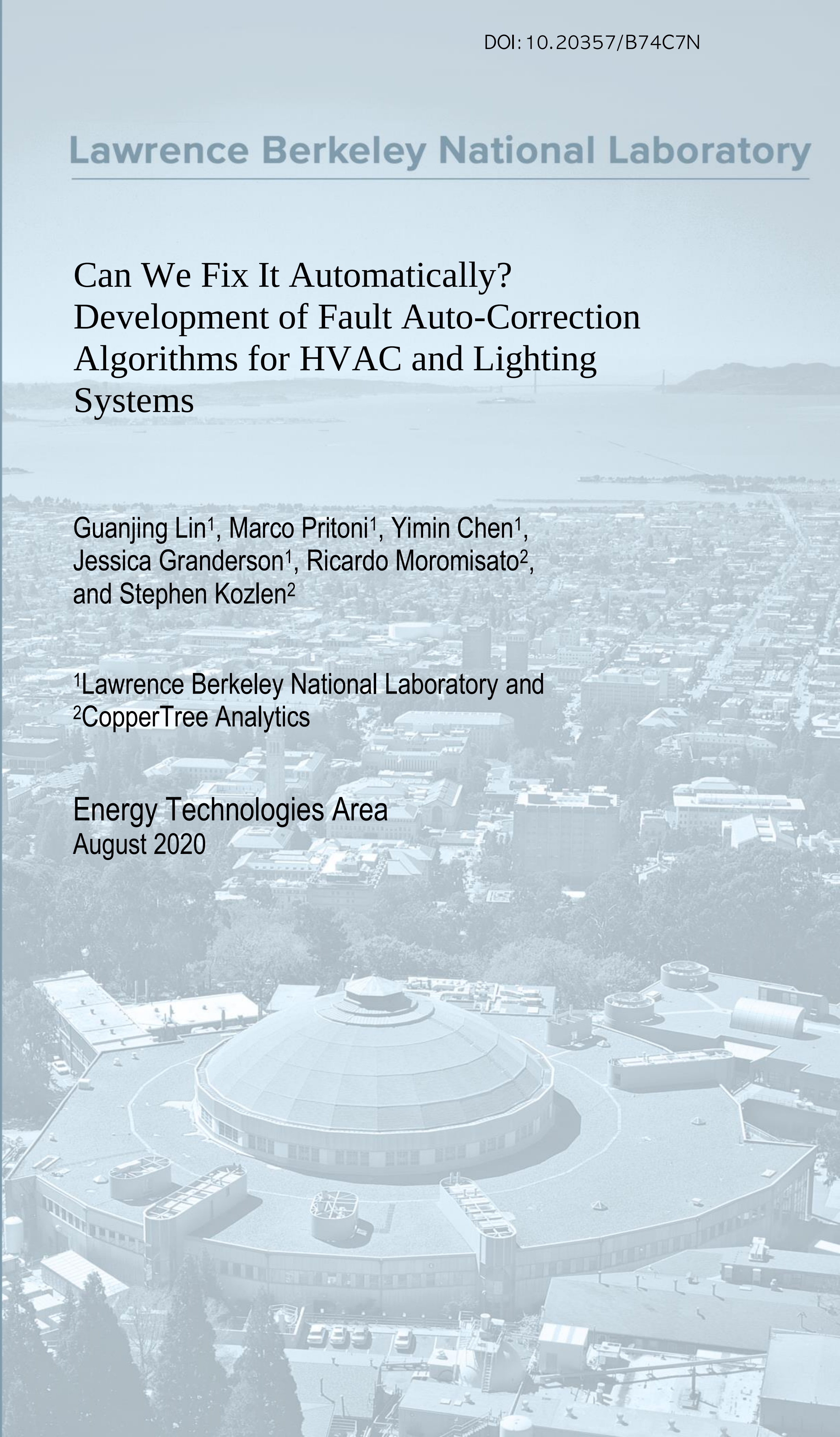
DOI: 10.20357/B74C7N

BERKELEY LAB

Lawrence Berkeley National Laboratory



Can We Fix It Automatically? Development of Fault Auto-Correction Algorithms for HVAC and Lighting Systems

Guanjing Lin[1], Marco Pritoni[1], Yimin Chen[1], Jessica Granderson[1], Ricardo Moromisato[2], and Stephen Kozlen[2]

[1]Lawrence Berkeley National Laboratory and
[2]CopperTree Analytics

Energy Technologies Area
August 2020

Can We Fix It Automatically? Development of Fault Auto-Correction Algorithms for HVAC and Lighting Systems

Guanjing Lin, Marco Pritoni, Yimin Chen, Jessica Granderson, Lawrence Berkeley National Laboratory
Ricardo Moromisato, Stephen Kozlen, CopperTree Analytics

ABSTRACT

A fault detection and diagnostics (FDD) tool is a type of energy management and information system designed to continuously identify the presence of faults and efficiency improvement opportunities through a one-way interface to the building automation system and application of automated analytics. Building owners and operators at the leading edge of technology adoption are using FDD tools to enable average whole-building portfolio savings of 8 percent. Although FDD tools can inform building operators of operational faults, currently a manual action is always required to correct faults and generate the associated energy savings. A subset of faults, however, such as biased sensors and manual override, can be addressed automatically, removing the need for operations and maintenance staff intervention. Automating this fault "correction" can significantly increase the savings generated by FDD tools and reduce the reliance on human intervention. Doing so is expected to advance the usability, as well as the technical and economic performance, of FDD technologies.

In this paper, we present the development of 10 innovative fault auto-correction algorithms for HVAC and lighting systems. When the auto-correction routine is triggered, it will overwrite the control setpoints or other variables (via BACnet or other protocol) to implement the intended changes. These algorithms are able to automatically correct the faults or improve the operation associated with an incorrectly programmed schedule, override manual control, sensor bias, control hunting, rogue zone, and less aggressive setpoints/setpoints setback. The paper will also discuss the implementation of the auto-correction algorithms in FDD software products.

Introduction

The literature indicates that 5 to 30 percent of commercial building energy use is wasted due to problems associated with controls (Katipamula and Brambley 2005; Roth, Westphalen et al. 2005; Fernandez, Katipamula et al. 2017; Deshmukh, Glicksman et al. 2018; Fernandes, Granderson et al. 2018). Fault detection and diagnostics (FDD) tools automatically identify building system or equipment-level operational issues, and in some cases are able to isolate the root causes of the problem. They use computer algorithms to continuously analyze system-level operational data to detect faults and diagnose their causes. Modern FDD tools typically integrate data from building automation systems (BAS). As depicted in Figure 1, integration with the BAS is implemented through a one-way interface, most commonly leveraging the BACnet protocol. Through this interface, system- or equipment-level operational data are made available to the FDD software. Data are continuously analyzed and detected faults are presented to operational staff through a graphical user interface. Since the BAS is the primary source of data, FDD is most commonly focused on heating, ventilation and air conditioning (HVAC) equipment.

However, today's technologies offer extensive libraries of FDD logic and algorithms, and they can be applied to lighting and other building end-use systems for which data are available (Granderson, Singla et al. 2017). FDD tools automate investigations that can be conducted via manual data inspection by someone with expert knowledge, thereby expanding accessibility and breadth of analysis opportunity, and also reducing complexity.

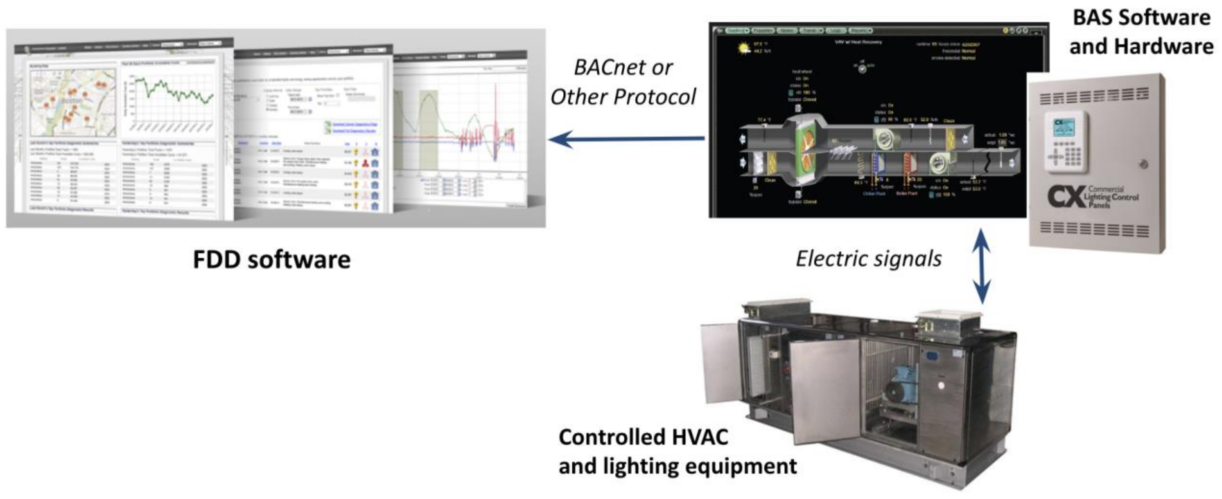

BAS Software and Hardware
BACnet or Other Protocol
FDD software
Electric signals
Controlled HVAC and lighting equipment

Figure 1. Schematic illustration of the integration of BAS data into FDD products

Today's FDD technology has been documented to enable whole building savings of 8 percent on average, across users (Lin, Kramer et al. 2020). Although FDD tools are being used to enable cost-effective energy savings, there is a capability gap in current products. Today's FDD technologies operate in an open loop manner. Faults are identified by the FDD tools. FDD tools may also provide a report of the duration and frequency of faults, cost and/or energy impacts, and relative priority levels. However, the identified faults are not automatically corrected. BAS and automated HVAC control optimization technologies offer closed loop supervisory control, but do not provide full-fledged, robust, continuous FDD. In practice, the need for human intervention to fix faults once they are identified often results in delay or inaction, resulting in additional operations and maintenance (O&M) costs or lost opportunity. This capability gap is not only technical, but also represents market-relevant desired functionality on behalf of FDD users and vendors (Crowe 2018; Granderson 2017). Therefore, this work seeks to develop automated fault correction approaches and integrate them with commercial FDD technology offerings, thereby closing the loop between passive diagnostics and active control. Automating the correction of these types of faults can increase the savings realized through the use of FDD tools, and reduce the extent to which savings are dependent upon human intervention.

The academic literature has extensively covered the development of automated FDD algorithms applied to HVAC and lighting systems (Kim, Rao et al. 2010; Shi and O'Brien 2019); however, very little has been published on the automatic correction of the identified faults via the actual control system. Fernandez et al. (Fernandez, Brambley et al. 2009a; Fernandez, Brambley et al. 2009b) and Brambley et al. (Brambley, Fernandez et al. 2011) developed passive and proactive fault auto-correction algorithms for an air-handler unit (AHU) and a variable-airvolume (VAV) box. The methods are proposed to correct faults occurring in temperature and humidity sensors, dampers, control hunting, and manual overrides. A subset of these algorithms (sensor bias and minimum outdoor air damper position) are implemented and tested in a laboratory experiment.

This paper presents a comprehensive set of auto-correction algorithms aimed at being integrated with commercial FDD tools and by describing how these routines are being integrated into existing FDD products through work with industry FDD providers. These algorithms target incorrectly programmed schedules, overriding manual control "lock out," sensor bias, control hunting, rogue zone, and suboptimal setpoints/setpoints setback. This paper also presents the new insights that are gained by integrating of these developed auto-correction algorithms into commercial FDD tools.

Fault Auto-correction Algorithms

Auto-correctable Faults of Focus

To identify the faults that are auto-correctable, we reviewed existing literature and discussed results with 10 subject matter experts with years of experience in FDD research and application. These experts included a set of FDD technology and service providers from the industry who are participating in this R&D effort as implementation partners. Those providers maintain a large footprint in the FDD market, and have explicitly included fault correction in their product development roadmaps. It is not possible to automate the correction of mechanical faults such as failed actuators, however, there is nonetheless a compelling set of operational problems that are detectable in today's FDD offerings, and correctable through software-based manipulation of the BAS parameters that can be exposed to external applications via BACnet.

Considering both the possibility of automated correction and feasibility of implementation in an FDD platform, a set of nine HVAC system faults and one lighting system fault were isolated (Table 1). For each of these faults, correction algorithms were then developed. The auto-correctable faults are divided into two categories: "Fault" and "Opportunity." Faults 1- 6 in Table 1 are in the "Fault" category which indicates problems that violate the intended operation of equipment (e.g., sensor bias). Faults 7-10 in Table 1 are in the "Opportunity" category which indicates problems that represent potential to improve the current operation of the equipment (e.g., improve setpoint reset). This distinction was made to differentiate between the intent of restoring operation to what it was intended to be and that of optimal control.

Table 1. Summary of the auto-correctable faults of focus in this study

Fault	**Fault description**
1. HVAC schedules are incorrectly programmed	AHU or other HVAC equipment does not turn on/off according to intended schedule due to an error in the control programming.
2. Override manual control	Operator unintentionally neglects to release what was intended to be a short-term override of set points or other control commands (e.g., fan VFD speed, cooling coil valve control command).
3. AHU outside air or supply air temperature sensor bias	AHU's outside air or supply air temperature sensor measurements have constant bias over time, representing an offset from a correct or true value.
4. Damper/valve/fan /pump control hunting	The damper, valve, pump, or fan hunting fault due to improper proportional gain.
5. Rogue zone	A zone that continuously sends cooling/heating requests whenever its schedule is on, due to zone-level equipment problems like a leaky reheat valve, a dysfunctional supply air damper, or an insufficient capacity VAV terminal.
6. Lighting schedules are incorrectly programmed	Lights do not turn on/off according to the intended schedule due to an error in the control programming.
7. Improve economizer high lockout temperature setpoint	In the AHU with fixed dry-bulb economizer control, the economizer shall be disabled whenever the outdoor air conditions exceed the economizer high lockout temperature setpoint. If the setpoint is set too low in the control logic, it will result in a missed opportunity to use outdoor air to reduce the mechanical cooling load in mild conditions.
8. Improve zone temperature setpoint setback	Each zone shall have separate occupied and unoccupied cooling and heating setpoints. If the room temperature cooling setpoint is too low or the heating setpoint is too high, the space will be over-cooled or overheated, causing unintended energy consumption.
9. Improve AHU static pressure setpoint reset	There is non optimized AHU static air pressure setpoint
10. Improve AHU supply air temperature setpoint reset	There is non optimized AHU supply air temperature setpoint

Overview of Auto-correction Algorithms

Although it may ultimately be desirable to develop an integrated auto-correction algorithm that fulfills the entire process of detection, diagnosis, and correction, in this study the primary objective is to develop automated fault correction algorithms that can be integrated with existing commercial FDD and BAS products. Therefore, the auto-correction algorithms described in this section are decoupled from the FDD algorithms embedded in the existing FDD tools. This permits applicability and feasibility of the developed correction algorithms across a variety of FDD technologies that employ different FDD rules and algorithms. Further, it is assumed that the FDD tools are able to detect the faults of focus, as they represent some of the more commonly encountered faults in commercial buildings.

Figure 2 shows a flow chart of the general auto-correction process. In this process, after the FDD algorithm generates a fault flag of a specific fault, the fault auto-correction algorithm is initiated to correct this fault with approval from the building operator. A correctible variable (referred to as Control_variable_being_overwritten), is the key element in the auto-correction process. The algorithm overwrites this variable (Control_variable_being_overwritten_current) to a new value (Control_variable_being_overwritten_new). The control_variable_being_overwritten_current is the one identified in the FDD algorithm to be associated with the problematic value (fault) or potential to improve (opportunity). The control_variable_being_overwritten_new is the same variable that has the correct value (fault) or optimized value (opportunity). All of the auto-correction algorithms developed in this work follow this structure, with different control variables overwritten in BAS, and different ways to determine the correct or improved value of the variable.

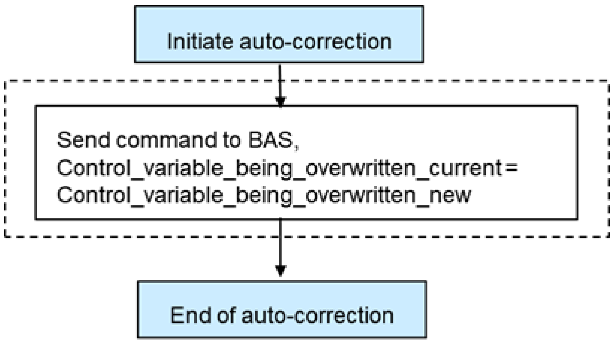

Initiate auto-correction
Send command to BAS,
Control_variable_being_overwritten_current =
Control_variable_being_overwritten_new
End of auto-correction

Figure 2. Flow chart of the general auto-correction process

Table 2 lists the correctible variables used in each algorithm that will be overwritten with the new value. Those variables can either be found in the BACnet object property library or created by the tool developers.

Table 2. Summary of the control variables overwritten in each auto-correction algorithm

Auto-correction Algorithm	**Control_variable_being_overwritten**	**Impleme -ntation Partner**
1. Schedules are incorrectly programmed	Equip_Schedule: HVAC equipment on/off times that are programmed in the BAS	Partner_3
2. Override manual control	Manual_Override: The control variable that indicates the equipment manual control status: 1 –in manual control, 0 – in automatic control	Partner_3, Partner_2
3. AHU outside air or supply air temperature sensor bias	TempSensorOffset: The offset of the controller's outside air or supply air temperature input or SAT_spt: Supply air temperature setpoint Eco_HighLimit: The economizer high lockout setpoint. The outside air temperature above which the economizer will return to the minimum position	Partner_1, Partner_2
4. Damper/valve/fan /pump control hunting	Kp: the proportional gain Kp in proportional–integral–derivative (PID) controller	Partner_3, Partner_1
5. Rogue zone	IM: Importance Multiplier of the rogue zone, or I: Number of Ignored Requests of the associated AHU	Partner_1
6.Lighting schedules are incorrectly programmed	Lighting on/off times that are programmed in the BAS	Partner_3
7. Improve economizer high lockout temperature setpoint	Eco_HighLimit	Partner_1, Partner_2
8. Improve zone temperature setpoint setback	ZAT_cspt_occ/ZAT_cspt_unocc: The zone temperature cooling setpoint during the occupied/unoccupied hours ZAT_hspt_occ/ZAT_hspt_unocc: The zone temperature heating setpoint during the occupied/unoccupied hours	Partner_3
9. Improve AHU static pressure setpoint reset	SSP_spt: Supply static pressure setpoint	Partner_1, Partner_2
10. Improve AHU supply air temperature setpoint reset	SAT_spt	Partner_1, Partner_2

The algorithms 1–8 in Table 2 correct the fault "one time." That is, the auto-correction completes when the new value of the control variable is implemented in the BAS. The auto-correction will not start again until another future instance of the fault is detected by the FDD algorithm (and correction is approved by the building operator). The algorithms 9 - 10 in Table 1 correct the fault "continuously" as it continuously adjusts control variables to optimize equipment operation (e.g., resets). These two algorithms are most closely related to optimal controls.

Examples of Auto-correction Algorithms

Two auto-correction algorithms are presented below as examples.

Example Algorithm 1. Schedules are incorrectly programmed

Building occupancy and HVAC equipment schedules are often found to be incorrectly programmed, or not as intended (Chen, Deng et al. 2002). This would cause a significant energy waste if the fault remains unattended and no correction is carried out.

In the fault auto-correction algorithm, it overwrites the Equip_Schedule to the Intended_Schedule. The Intended_Schedule is prior knowledge that is specified by the building operator. Depending upon BAS implementation, the equipment status is controlled by Equip_Schedule at the equipment level or the master schedules programmed in the supervisory level. Therefore, it is necessary to investigate where the Equip_Schedule object is located in the actual building implementation. Figure 3 illustrates the algorithm's working flow.

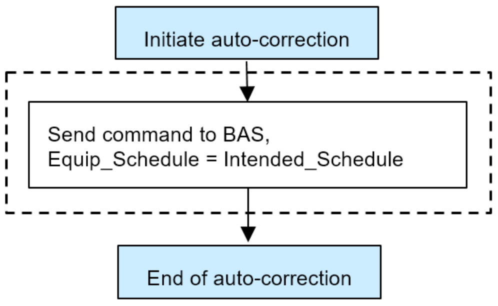


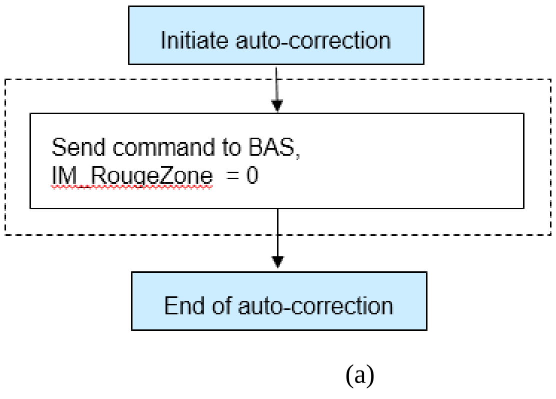
Figure 3. Flowchart of "schedules are incorrectly programmed" fault auto-correction algorithm

Example algorithm 2. Rogue zone fault

A rogue zone is a zone that continuously sends cooling/heating requests whenever its schedule is on, due to the zone-level equipment problems such as a leaking reheat coil valve, a malfunctioning supply air damper, an insufficient capacity VAV terminal, and others. Excluding rogue zones from the corresponding Trim and Respond reset control strategies improves operation and increases energy savings (Peffer, Pritoni et al. 2016).

Two correction strategies were developed to eliminate the rogue zone impacts (ignore the cooling request from the rogue zone). The first is to overwrite the Importance Multiplier of the rogue zone. The Importance Multiplier is usually used in the control sequence to decide if the cooling requests or heating requests from the zone level should be used to control the upstream systems (e.g., the AHUs). When the FDD tool flags the rogue zone fault, the Importance

Multiplier of the rogue zone is overwritten to be zero. Therefore, the cooling requests or heating requests from the rogue zone can be removed. Figure 4a illustrates the algorithm's working flow.



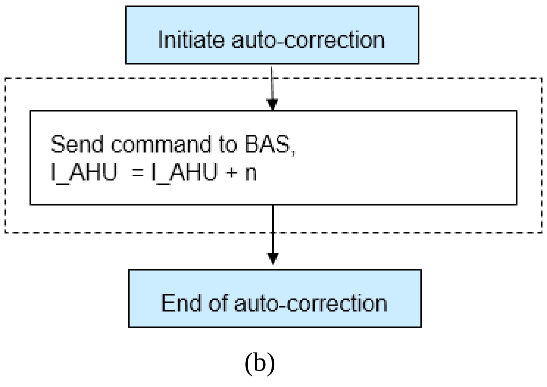


Figure 4. Flowchart of "rogue zone" fault auto-correction algorithm, (a) approach 1, (b) approach 2

The second is to overwrite the Number of Ignored Request points of the associated AHU in the BAS. When there are multiple zones calling cooling requests or heating requests, the auto-correction algorithm will calculate the number of requests based on predefined rules. Then the Number of Ignored Request can be increased by *n* in the BAS. The upper stream system will respond to the real zone requests after removing the number of ignored requests. The value of *n* is the same as the number of cooling requests determined in the control sequence of that rogue zone. Since *n* may change with the changes of zone condition (e.g., if the zone temperature exceeds the cooling setpoint by 5°F for two minutes, n = 3; else if the zone temperature exceeds the cooling setpoint by 3°F, n = 2; else if the cooling loop is greater than 95 percent, n = 1 until the cooling loop is less than 85 percent; else if the cooling loop is less than 95 percent, n = 0), using this approach needs to perform the auto-correction continuously. Figure 4b illustrates the algorithm's working flow.

Implementation of Auto-correction into the FDD Software

Three commercial FDD providers have integrated these routines into their development product environments for future field testing (Table 2). This section details the algorithm selection and implementation process.

Selection of the Fault Auto-correction Algorithm

Partner_1 selected algorithms 5, 9, and 10 in Table 2 because they want to optimize the operation of air handlers without programming complex sequences into the BAS controllers. This is particularly valuable since some of the underlying control infrastructure in buildings served by this partner is obsolete and heterogeneous. The selected algorithms impede cost-effective deployment of more efficient control algorithms. The FDD tool, enhanced with two-way communication, is planned to be used as a new supervisory controller over the existing BAS. In addition to supporting the implementation of algorithms 9 and 10, algorithm 5 is also applicable to many other buildings with minimal changes to the BAS. Partner 1 also selected algorithm 4, because many PID loops in their buildings are out of tune, and it would take significant operator time to fix them manually. Algorithm 3 was selected to test the potential of using a proactive diagnostic.

Partner_2 chose algorithm 2, since widespread and improper use of manual override is a very common issue for their clients. The selection of algorithm 3 and 9 was driven by the large savings expected by improving economizer operation, while the selection of algorithms 9 and 10 was driven by savings potential and the large number of potential sites in the portfolio.

Partner_3 selected auto-correction algorithms that could be applied to the majority of the buildings and BASs in their portfolio. They chose the first two algorithms and algorithm 6 as they built upon features their FDD platform already included. Their tool already benchmarks and stores several parameters describing the intended operation of the buildings, including schedules, control modes, and setpoints. These "optimal" parameters are continuously compared with current schedules and manual overrides, and when these settings are deviated from optimal ones, the facility managers are notified. However, if the users want to revert changes to optimal settings, they have to use the BAS front end. Partner_3 believes the new fault correction algorithms integrated into the FDD platform will empower users to take action using further automation. Automated PID loop tuning (algorithm 4) is a high-value commissioning feature for Partner_3, because unstable processes account for a large number of faults. This partner also selected algorithm 8, since continuously improving zone temperature setpoint is an extension of the other scheduling and override algorithms developed.

Implementation Process

Three commercial FDD providers implemented the selected fault auto-correction algorithms in their development product environment. The general activities in the implementation among the three partners include:

- Confirm/add two-way communication functionality between the FDD and the BAS.
- Build an auto-correction interface to communicate with the building operator.
- Translate algorithms to the code environment:
  - Understand control details to determine which specific control variables are related to the auto-correction.
  - Confirm/enable auto-corrective functionality on the control variables, as per above.
  - Determine the value of the control variable to be auto-corrected.
  - Map points to read from and write to.
  - Program the auto-correction routine in the analytics engine.
- Commissioning
  - Verify that the user can trigger the auto-correction command from the designed interface.
  - Test the algorithms, review the auto-correction algorithm outputs, and verify the point value to be auto-corrected.

Implementation Challenges and Solutions

This section describes a number of challenges that were faced during the implementation of auto-correction algorithms into the FDD tools, as well as the solutions that were used by one or more project partners to mitigate them.

Two-way communication between the FDD and the BAS

Establishing two-way communications between the FDD system and the BAS is a challenge seen across project partners implementing fault auto-correction into their FDD product environment. The current state-of-art FDD system only has a one-way communication capability with the BAS. In other words, it reads operational data from the BAS, runs analytics, and flags faults on the software interface. There are no capabilities to write or execute auto-corrective commands directly to the BAS. Therefore, opening two-way communication between the FDD system and the BAS is the first challenge encountered during the process of auto-correction implementation.

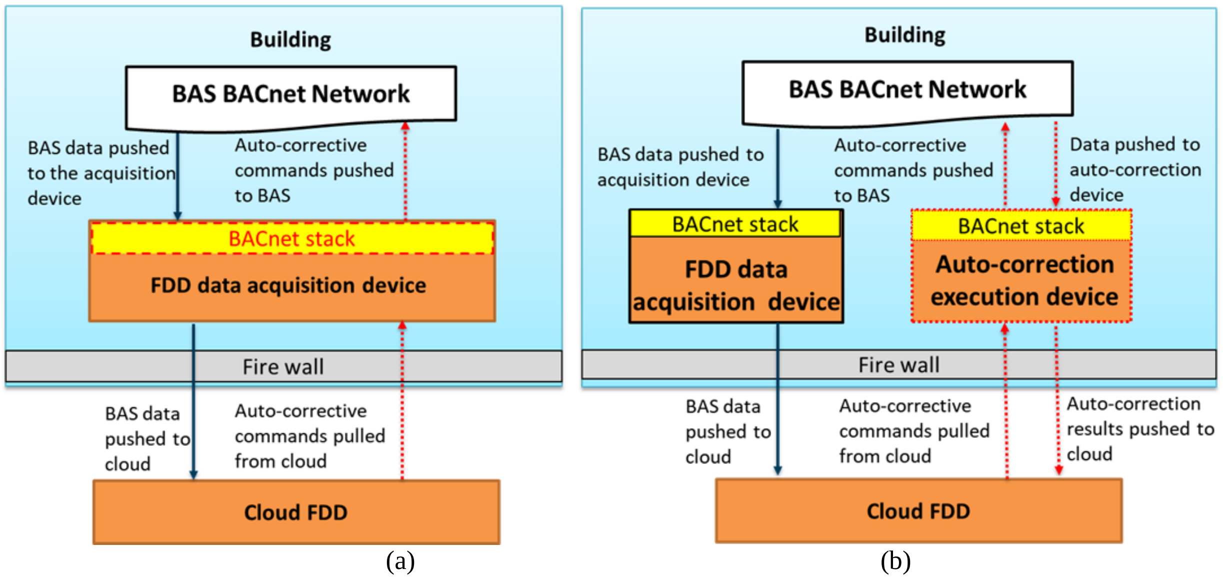

Building
BAS BACnet Network
BAS data pushed to the acquisition device
Auto-corrective commands pushed to BAS
BACnet stack
FDD data acquisition device
Fire wall
BAS data pushed to cloud
Auto-corrective commands pulled from cloud
Cloud FDD
(a)
Building
BAS BACnet Network
BAS data pushed to acquisition device
Auto-corrective commands pushed to BAS
Data pushed to auto-correction device
BACnet stack
FDD data acquisition device
BACnet stack
Auto-correction execution device
Fire wall
BAS data pushed to cloud
Auto-corrective commands pulled from cloud
Auto-correction results pushed to cloud
Cloud FDD
(b)

Figure 5. Two-way communication infrastructure for (a) the cloud-based FDD system 1, and (b) the cloud-based FDD system 2.

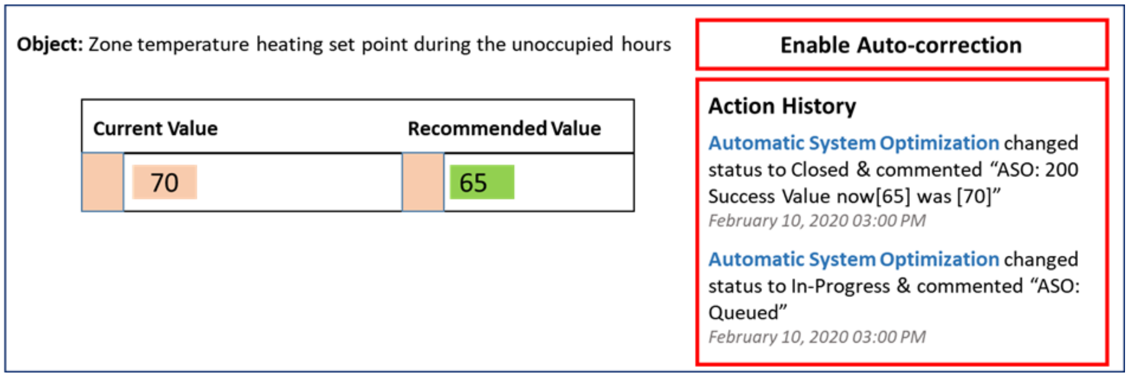
The project partners mitigated the two-way communication challenge by upgrading their FDD system infrastructure. Figure 5 illustrates the solutions for the two cloud-based FDD systems. The solid line shows the original infrastructure, and the red dashed line shows the upgrade. In the first cloud-based FDD case (Figure 5a), the BACnet stack (a software library allows users to add a native BACnet interface to talk to the devices or applications in the BACnet network) of the FDD data acquisition device was updated to include a “write” function. The local data acquisition device was also updated to make API requests to the cloud FDD platform to retrieve the auto-correction command information. This enabled the FDD system to send the auto-corrective command to the local device and then to the writable properties used to control the BAS. In a cloud-based FDD system of another partner (Figure 5b), the current

BACnet stack has the write function already. For two-way communication, a new field device (auto-correction execution device) was installed in the building in addition to the existing FDD data acquisition device. The cloud FDD engine initiates the auto-corrective command onto the auto-correction execution device. The device then executes the commands onto the BAS BACnet network and reports back the results to the cloud FDD engine. The BAS data are still acquired by the existing FDD data acquisition field device and delivered to the cloud FDD engine.

Auto-Correction Acceptability

The second challenge is building operator acceptance. The new auto-correction feature affords the FDD technology a certain degree of control capability. Building operators may be hesitant to trust this new capability and concerned that the BAS may lose its autonomy.

To mitigate the challenge of auto-correction acceptability, one project partner updated its existing user interface to make sure users are allowed to actively start, interrupt, and track the auto-correction activities. Auto-correction enable and disable functionality were added to the user interface (UI). To help building operators make the decision to start the auto-correction, the auto-correction details—such as the name of the control variable to be overwritten, its current value, and the value after the execution of the auto-correction—were also provided to them in the UI. All user and system activities related to auto-correction were stored in an action history log which can be retrieved easily by the operator to understand all past activities. Figure 6 shows a new UI example displaying the auto-correction enable/disable functionality, details, and the action history.


Object: Zone temperature heating set point during the unoccupied hours

Current Value	Recommended Value
70	65

Action History

Automatic System Optimization changed status to Closed & commented "ASO: 200 Success Value now[65] was [70]"
February 10, 2020 03:00 PM

Automatic System Optimization changed status to In-Progress & commented "ASO: Queued"
February 10, 2020 03:00 PM

Figure 6. New UI example displaying the auto-correction enable/disable functionality, details, and action history

Specifying and Accessing the Proper Overwritten Control Variable

The third challenge was to specify and access the proper control variable to be overwritten in buildings. Although the Control_variable_being_overwritten in each auto-correction algorithm is listed in Table 2, the name of the control variable varies depending upon specific BAS and building. Therefore, it is a challenge to specify the proper control variable. For instance, when implementing the "Override manual control" algorithm, it is necessary to understand which data point in the BAS represents the override. Depending upon the BAS, there are multiple ways to do the override. The override can be done via a Manual_Override data point whose value is: 1 – equipment is in manual control, 0 – equipment is in automatic control. Or the override may be done by setting the priority level of the BACnet writable points (e.g., 8 – manual operator override, 16—default automatically operation). In the former case, the Control_variable_being_overwritten should be the Manual_Override, and its value should be changed from 1 to 0 after auto-correction. But in the latter case, the Control_variable_being_overwritten should be the priority level, and its value should be changed from 8 to 16 after auto-correction.

Accessing the proper control variable is another part of the challenge. The auto-correction algorithms may require the FDD tools to have access to the control variables that are not commonly exposed by the BAS. An example is the PID tuning parameters required by the "control hunting" algorithm. PID tuning parameters are defined as optional by the BACnet standard. Often, they are set as proprietary by the BAS. If that is the case, the auto-correction is not possible as those variables are not able to be read and/or written.

To address the challenge of specifying and accessing the proper control variable, more details about the BAS and specific control configuration of the targeted test buildings were collected and the auto-correction algorithms were adapted accordingly. The identified control variables were examined if they were accessible. The parameters of a BAS controller, gateway, or server were changed to expose the proprietary points to make them readable and writable by the FDD tool via BACnet or a vendor API. Data types of the targeted control variable were also investigated. The algorithms were adjusted to allow them to write in the correct data format: (1) Binary property type: 2-state conditions represented as either "0" or "1", "false" or "true," "inactive" or "active"; (2) Analog property type: numbers that have different precision depending on the number of decimals used and the different variable types (e.g., float 32-bit precision, double 64-bit precision); and (3) Array property type: more complex data types such as schedule event arrays that have different data formatting representing, for example, a seven-day schedule with a different first day of the week (i.e., Sunday versus Monday).

Asynchronous Operation between the FDD and the Auto-correction

Asynchronous operation between fault detection and auto-correction execution is another problem to resolve. There might be a time delay between the time a fault is detected, the time it is approved by the building operator for auto-correction, and the time the auto-correction command is retrieved for execution. Therefore, inconsistencies may be observed in the control variable's value when the fault is detected and when the auto-correction command is executed.

To solve the problem and avoid the unintended impact, one partner implemented validation before the execution of auto-correction. The current value of the targeted control variable is read from the BAS and checked to see if it is still the value upon which the fault condition was detected. If the validation fails, the auto-correction command is deemed invalid and is not executed at the BAS.

Conclusion and Future Work

Although FDD tools are commercially available and being used to improve efficiency, there is a capability gap in current products. Automating the correction of faults closes the loop between passive FDD and active control, increases the savings realized through the use of FDD tools, and reduces the extent to which savings are dependent upon human intervention. This paper presents 10 algorithms for HVAC and lighting systems that are designed to automatically correct faults or improve operations relative to incorrectly programmed schedules, overriding manual control, sensor bias, control hunting, rogue zones, and less aggressive setpoints/setpoints setback. It also illustrates the findings obtained from the integration of these auto-correction algorithms into three commercial FDD tools for future field testing. The FDD service providers provides selected specific algorithms to integrate into their tool, considering the savings potential of the faults, the common issues of their clients, and the required changes to the BAS and the FDD tool, among others. The key activities of the integration consist of confirming/adding two-way communication functionality, creating an auto-correction interface for communication, translating algorithms to the code environment, and commissioning the integrated solution.

Although the algorithms have been successfully integrated into the FDD tools, a number of practical challenges were faced during the integration, and those are highlighted in this paper. The challenges include two-way communication between FDD and BAS, auto-correction acceptability, specifying and accessing the proper overwritten control variable, and asynchronous operation between FDD and auto-correction. For each challenge, solutions utilized by one or more project partners are described. These suggested solutions will help future auto-correction developers address similar challenges.

Future work will focus on field testing the FDD integrated correction algorithms in a cohort of existing buildings. This will include evaluation of the technical efficacy and performance of each correction routine, evaluation of the operations and maintenance benefits as well as energy savings for each site in the cohort, and characterization of challenges and best practices. A second area of future work will entail design and execution of a techno-economic analysis to quantify the broader market opportunity to inform ongoing commercialization efforts.

Acknowledgement

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Office, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. The authors wish to acknowledge Harry Bergmann for his guidance and support of the research. We also thank the fault detection and diagnostics technology and service providers who participated in this study.

References

Brambley, M. R., N. Fernandez, W. Wang, K. A. Cort, H. Cho, H. Ngo and J. K. Goddard. 2011. “Final Project Report: Self-Correcting Controls for VAV System Faults Filter/Fan/Coil And VAV Box Sections”. Report PNNL-20452. Pacific Northwest National Lab.(PNNL), Richland, WA (United States). https://buildingsystems.pnnl.gov/documents/PNNL-20452%20Self-Correcting%20Controls%20Final%20Report%202011.pdf.

Chen, H., S. Deng, H. Bruner, D. Claridge and W. Turner. 2002. “Continuous Commissioning Results Verification and Follow-up for an Institutional Building: A Case Study”. Symposium on Improving Building Systems in Hot and Humid Climates.

Crowe, E, Karmer, H, Granderson, J. “Summary of Outcomes of 2018 Smart Building Roundtable Workshop”. 2018. Better Buildings Alliance EMIS R&D Team. Lawrence

Berkeley National Lab., Building Technologies and Urban Systems Division; http://eis.lbl.gov/pubs/SB-roundtablesummary.pdf.

Deshmukh, S., L. Glicksman and L. Norford. 2018. "Case Study Results: Fault Detection in Air-Handling Units in Buildings." *Advances in Building Energy Research*: 1-17.

Fernandes, S., J. Granderson, R. Singla and S. Touzani. 2018. "Corporate Delivery of a Global Smart Buildings Program." *Energy Engineering* 115(1): 7-25.

Fernandez, N., M. R. Brambley and S. Katipamula. 2009a. "Self-correcting HVAC Controls: Algorithms for Sensors and Dampers in Air-Handling Units". Report PNNL-19104. Pacific Northwest National Lab.(PNNL), Richland, WA (United States). https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-19104.pdf.

Fernandez, N., M. R. Brambley, S. Katipamula, H. Cho, J. Goddard and L. Dinh. 2009b. "Self-Correcting HVAC Controls Project Final Report". Report PNNL-19074. Pacific Northwest National Lab.(PNNL), Richland, WA (United States). https://www.pnnl.gov/main/publications/external/technical_reports/PNNL-19074.pdf.

Fernandez, N. E., S. Katipamula, W. Wang, Y. Xie, M. Zhao and C. D. Corbin. 2017. "Impacts of Commercial Building Controls On Energy Savings And Peak Load Reduction". Report PNNL-25985. Pacific Northwest National Lab.(PNNL), Richland, WA (United States). https://buildingretuning.pnnl.gov/publications/PNNL-25985.pdf.

Granderson, J., R. Singla, E. Mayhorn, P. Ehrlich, D. Vrabie and S. Frank. 2017. "Characterization and Survey of Automated Fault Detection And Diagnostic Tools." Report LBNL-2001075, Lawrence Berkeley National Laboratory, Berkerly, CA.

Katipamula, S. and M. R. Brambley. 2005. "Methods for Fault Detection, Diagnostics, And Prognostics for Building Systems—a Review, part I." *HVAC&R Research* 11(1): 3-25.

Lin, G., H. Kramer and J. Granderson. 2020. "Building Fault Detection and Diagnostics: Achieved Savings, and Methods to Evaluate Algorithm Performance." *Building and Environment* 168: 106505.

Peffer, T., M. Pritoni, G. Fierro, S. Kaam, J. Kim and P. Raftery. 2016. "Writing Controls Sequences for Buildings: From HVAC Industry Enclave to Hacker's Weekend Project". ACEEE Summer Study on Energy Efficiency in Buildings, Asilomar, CA.

Roth, K. W., D. Westphalen, M. Y. Feng, P. Llana and L. Quartararo. 2005. "Energy Impact Of Commercial Building Controls and Performance Diagnostics: Market Characterization, Energy Impact of Building Faults and Energy Savings Potential." Prepared by TAIX LLC for the US Department of Energy. November. 412pp (Table 2–1).